Context matters: Evaluating Interaction Techniques with the CIS Model

Caroline Appert, Michel Beaudouin-Lafon, W.E. Mackay

Context matters: Evaluating Interaction Techniques with the CIS Model

This article introduces the Complexity of Interaction Sequences model (CIS). CIS describes the structure of interaction techniques and predicts their performance in the context of an interaction sequence. The model defines the complexity of an interaction technique as a measure of its effectiveness within a given context. We tested CIS to compare three interaction techniques: fixed unimanual palettes, fixed bimanual palettes and toolglasses. The model predicts that the complexity of both palettes depends on interaction sequences, while toolglasses are less context-dependent. CIS also predicts that fixed bimanual palettes outperform the other two techniques. Predictions were tested empirically with a controlled experiment. We argue that, in order to be generalisable, experimental comparisons of interaction techniques should include the concept of context sensitivity.

Keywords: Interaction technique, Interaction Sequence, Complexity, Context, Palette, Bimanual Palette, Toolglass, Experimentation, Performance, Theory.

1 Introduction

Research in HCI has produced many novel interaction techniques aimed at improving the usability of graphical applications. Yet very few make it into real products. This may be due to the difficulty of assessing the actual value of a technique before it is integrated into a real interface. Researchers often evaluate new interaction techniques with usability studies. However, the results from these

C. Appert, M. Beaudouin-Lafon, and W. Mackay. (2004). Context matters : Evaluating interaction techniques with the CIS model. In Proc. People and Computers XVIII - Design for Life -, HCI 2004, Leeds, UK. Springer Verlag, Sept. 2004, pages 279-295.

studies are often specific to the software and setting, making them hard to generalise. An alternative is a controlled experiment that measures the performance of the technique using a benchmark task. However the choice of the task is crucial: the designers of the technique have an incentive to create test tasks that optimise performance of the technique, as opposed to evaluating its actual performance in context.

How can we capture the context of use to better evaluate an interaction technique? We introduce a new model, Complexity of Interaction Sequences (CIS), that addresses context through the notion of an interaction sequence. This is based on the observation that users organise their interactions according to their cognitive context. For instance, in a copy context, users tend to create objects of the same type in sequence whereas in a problem solving context, they create objects according to their thought process (Mackay, 2002). Even though the two interaction sequences may lead to the same result, a given interaction technique may perform better with one sequence than the other. Therefore, the evaluation of interaction techniques must take into account the various contexts in which they may be used.

The CIS model introduced in this paper was designed to *describe* the structure of interaction techniques, *analyse* them through a set of criteria and to *measure* the complexity of an interaction technique in order to *predict* its effectiveness given a particular interaction sequence. The goal of CIS is to complement other evaluation techniques by helping researchers understand the effect of context on the performance of interaction techniques.

After a review of related work, we present the CIS model and apply it to three techniques: fixed unimanual palettes, fixed bimanual palettes and toolglasses. We use CIS to understand how these techniques are sensitive to context and test these predictions with a controlled experiment. We conclude with directions for future work.

2 Related Work

Few controlled studies have attempted to explicitly take context of use into account. For example, toolglasses (Bier et al., 1993) are semi-transparent movable tool palettes used with two hands. To apply a tool to an object, the user clicks through the tool onto the object. Kabbash et al. (1994) report that toolglasses are faster than other palettes. However, the benchmark task used favours toolglasses because it forces the user to always select a different tool. Selecting a new tool requires a round trip to the palette, while the toolglass is always at hand. Even though the experiment was properly controlled and the results carefully reported, it is not clear that the results can be generalised.

Generalising the results of such controlled studies requires a better understanding of the influence of context of use on performance. In the Cognitive Dimensions Framework, Green (2000) defines six types of activities such as transcription and incrementation and a set of dimensions such as viscosity and visibility to evaluate information artifacts. He shows that users adapt their behaviour to the type of activity and identifies the most important dimensions for each activity. CIS is

influenced by this framework that address the interplay between the task at hand and the properties of the available interaction techniques.

Mackay (2002) compares the efficiency of three interaction techniques (toolglasses, floating palettes and marking menus) used in the CPN2000 interface (Beaudouin-Lafon & Lassen, 2000) according to two cognitive contexts: copy and problem solving, similar to Green's transcription and incrementation. She observes that the use of a tool varies according to the context as well as users' preference. Users' preference and efficiency are higher with floating palettes in a copy context and with toolglasses or marking menus, i.e. circular contextual menus augmented by a gesture recognition mechanism, in a problem solving context. In other words, which technique is "best" depends on the context of use.

Unfortunately, such controlled experiments are costly and it would not be practical to test all possible tasks. What is needed is a model that can describe interaction techniques and predict their comparative performance in realistic settings. Formal models of interaction are too numerous to be reviewed exhaustively here. We focus on those that address interaction at a level of abstraction similar to CIS.

Card et al (1991) introduce a taxonomy of input devices, described as translators from physical properties to logical parameters of an application. Input devices are evaluated by their expressivity and efficiency, as measured by pointing speed and precision, footprint, etc. CIS analyses interaction at a higher level than input devices and elementary tasks by focusing on interaction techniques.

Goals, Operators, Methods and Selection rules (GOMS) (John & Kieras, 1996) is a family of descriptive and predictive models based on task analysis. Keystroke-Level Model (KLM) describes a task as a totally ordered sequence of operators while CMN-GOMS, NGOMSL and CPM-GOMS describe a task as a hierarchy of goals with operators as leaves. A goal can be reached by a method, described as a sequence of sub-goals and operators. Selection rules are ad hoc rules to choose a method when a goal can be reached by several methods (for instance, if the goal is “selecting text” and the text is composed of one word then double click else press mouse button at the beginning of the text and move to the end of the text). KLM and CMN-GOMS predict the time needed to achieve a task by summing the times required by each operator in the sequence (for instance, the operator P, Pointing, requires 1.10 s.). NGOMSL and CPM-GOMS are more elaborate models based on cognitive theories. Using Cognitive Complexity Theory (Kieras & Polson, 1985), NGOMSL can predict the time required to learn a method; Using the Model Human Processor (Foley et al., 1984), CPM-GOMS predicts how highly skilled users will perform several operators in parallel. Despite tools such as Apex (John et al., 2002) that automate part of building a CPM-GOMS model, constructing a model can be quite hard and long.

Like many of these models, the goal of CIS is to assess performance by predicting execution times. However the approach is different from GOMS: we focus on user interaction rather than tasks and on sensory-motor rather than cognitive aspects of interaction.

3 Describing Interaction Techniques with CIS

The CIS model is based on the description of interaction techniques: rather than describing user tasks (how the system is used), we describe the interface (what the system has to offer). Our hypothesis is that the "details" of interaction have a significant impact on performance, especially for skilled users who optimise for time, and therefore the sensory-motor aspects of interaction are critical to accurately predict execution times. We are not interested in the cognitive aspects of interaction per se, such as how users plan their tasks, but how the results of such planning affect performance. Our notion of interaction sequence captures a "trace" of the cognitive process and is all we need to model the context of use.

3.1 Defining an Interaction Technique

CIS describes an interface as a set of objects that users can manipulate. Some objects are *work objects*, e.g. geometric shapes, while others are *tool objects*, e.g. menu items and toolbars. The *state of the interface* is defined by the set of work and tool objects and the values of their attributes. A *manipulation* is a creation, modification or deletion of work objects. It is described by a tuple of the form (command, attributes). The *interaction space* is the set of manipulations available to the user in a given interaction state. An *interaction step* is a sequence of *actions* that progressively reduce the interaction space to a single manipulation that it executes, leading to a new state and therefore a new interaction space.

An *interaction technique* is a set of interaction steps. CIS describes it with an oriented graph, called the *interaction graph*. Figure 1 shows the interaction graph for traditional fixed palettes in a simple interface that can create rectangles, ellipses and triangles of a predefined size.

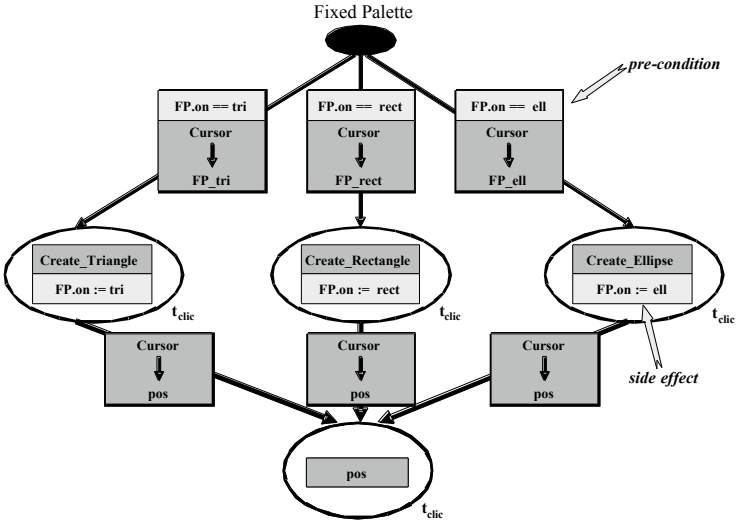


Figure 1: *Interaction graph* for a Fixed Palette.

The root is labeled with the technique name. A path from the root to a leaf models an interaction step. For example, the leftmost path in the graph (Figure 1) describes the following sequence of actions: move the cursor over the triangle tool – click – move the cursor over a position – click. We distinguish two types of actions, *acquisitions* and *validations*, described by arcs and nodes in the interaction graph:

- An *acquisition* (arc) identifies a subset of the current interaction space; it is usually achieved by moving an object, typically the cursor, over a tool, a work object or a position. In the interaction graph, the arc is labeled by the object being moved and its target.
- A *validation* (node) confirms the subset identified by an acquisition, which becomes the current interaction space; it is usually achieved by clicking a button or typing a key. In the interaction graph, each node (except for the root) is a validation, labelled by the element(s) of the manipulation it instantiates and the duration of the physical action

The leftmost path in figure 1 first instantiates the *c* (command) field of the manipulation with *Create_Triangle* when moving the cursor on the tool that creates rectangles, reducing the interaction space to the set {(*Create_Triangle*, p) | p ∈ position}. It then instantiates the *p* (position) field when clicking at position *pos*, reducing the interaction space to the single manipulation (*Create_Triangle*, *pos*) and executing it.

Nodes can have side effects that describe the change of state of the interface other than the changes of object positions. For example, selecting a tool in a palette activates this tool for future actions. Arcs can have preconditions: when true, the acquisition and validation actions are skipped. For example, if the triangle tool is already selected, the first step is skipped and a single click on the desired position creates a new triangle.

3.2 Properties of Interaction Techniques

Interaction graphs can describe a large variety of interaction techniques. We have found the following set of criteria both easy to apply and useful to compare the techniques qualitatively:

- *Order* and *Parallelism*

An interaction technique imposes a sequential and/or parallel organisation of its constituent actions, visualised by the shape of the interaction graph and the use of the parallel construct. For example, the interaction graphs for toolglasses and fixed palettes (Figures 1 and 2) show that a toolglass is highly parallel while a palette is highly sequential.

- *Persistence*

Interaction techniques may have side effects such as setting attributes of tool objects. These side effects may affect how the interaction technique is used the next time, as described by the pre-conditions in the interaction graphs. For example, the tool selected when using a traditional palette is persistent, so, for example, creating two rectangles in a row only requires selecting the rectangle tool once. Toolglasses on the other hand, do not have such persistence.

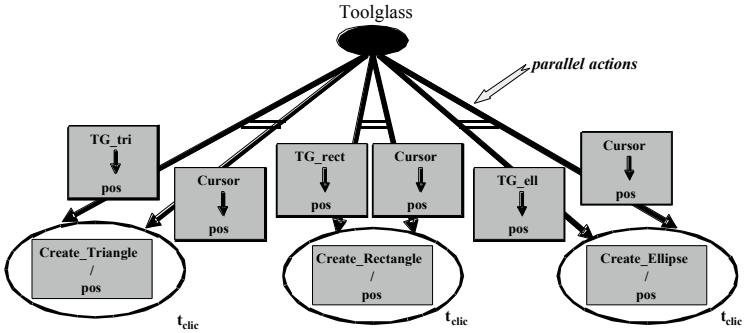


Figure 2: *Interaction graph* for a Toolglass.

- *Fusion*

Some interaction techniques can modify the several work objects by specifying multiple manipulations at once. For example, many drawing tools support the acquisition of several shapes by pressing the SHIFT key and modifying them all at once. Other tools use integrality principles (Jacob et al., 1994) to manipulate multiple attributes of an object at once, such as the style and thickness of lines in a drawing editor.

- *Development*

Some interfaces allow the user to create several copies of a tool with different values for its attributes. For example, in HabilisDraw (Amant & Horton, 2002), multiple *ink wells* can be used to colour objects. This is more efficient than using a single colour tool and changing its colour, but uses more screen real estate.

4 Making Predictions with CIS

4.1 Predictive Power: the Complexity Measure

In order to measure the efficiency of an interaction technique, we introduce a *measure of complexity*, inspired by the measure of complexity used in evaluating algorithms. An *interaction sequence* is a sequence of interaction steps that changes the state of the interface. We define a *problem* to be solved as a state to be reached using an interaction sequence. The *size* of the problem is the length of the sequence. The *actions* are the acquisition and validation actions used in an interaction sequence that solves the problem, i.e. which activate manipulations in the sequence. The *complexity* of an interaction technique for the given sequence measures the cost of the actions relative to the size of the problem when using this interaction technique. We use two measures: the number of actions to solve the problem and the execution time of these actions. Figure 3 shows how several interaction sequences can solve equivalent problems, i.e. reach the same state for work objects. As with algorithms, we can explore the best- and worst-case complexities, i.e. the interaction sequences that solve equivalent problems with the lowest and highest values.

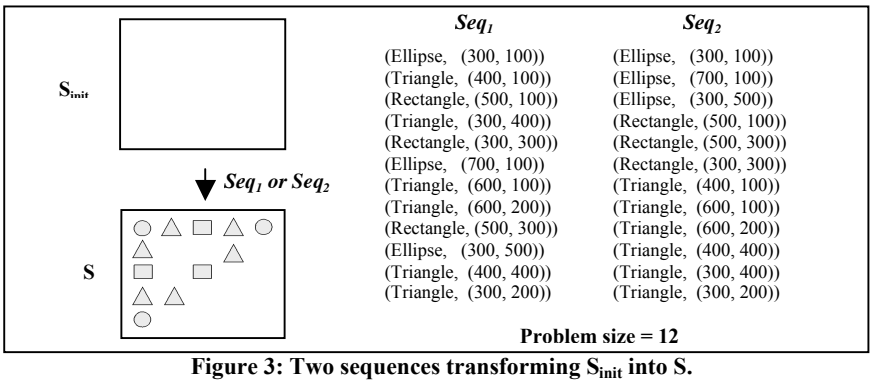


Figure 3: Two sequences transforming S_{init} into S.

We have developed an application, SimCIS, that simulates the use of an interaction technique and predicts its complexity. It takes as input :

- the initial state of the interface, S_{init}.
- the *interaction graph* of a technique, IG_{tech}.
- the *interaction sequence*, *Seq*.

SimCIS constructs the *sequence graph* that describes the overall interface by merging together the roots of all the interaction graphs and adding arcs from each leaf to the new root. Any path starting and ending at the root of the sequence graph instantiates an interaction sequence (such paths will typically go through the root multiple times). SimCIS computes the path *P* that activates the manipulations of the sequence *Seq* and evaluates the action and time complexity (Figure 4).

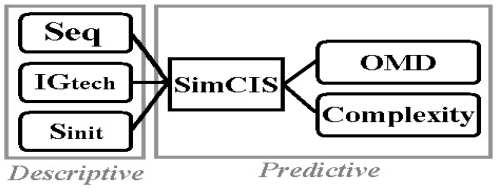


Figure 4: the CIS Model.

Action complexity is computed as the number of nodes and the number of arcs in the path *P*. When a pre-condition is true, the corresponding arc and end node are not counted. Time complexity is computed by summing the time taken by each arc and node in *P*. The time taken by an arc is the sum of the time taken to choose that arc at the parent node and the time taken by the pointing action. The former is estimated by Hick's (1952) law which models the choice selection time: $k \log_2(1+n)$ for *n* arcs; the latter is estimated by Fitts' (1954) law which models the pointing time: $a + b \log_2(1+D/W)$ for a target of size *W* at distance *D*. The time taken by a node is the constant time that labels the node. When the pre-condition of an arc is true, the time of the arc and its end node are ignored. We have used the following values, taken from the literature, for the constants in Hick's and Fitts' law: k = 150, a = 0 ms, b = 100.

SimCIS also generates a diagram illustrating the different object movements for the sequence, called the Object Movement Diagram (OMD, see Figure 6). These

diagrams make it easy to analyse and compare the performance of interaction techniques. The vertical axis represents time (downward) while the horizontal axis approximates the distances between objects. Movable objects and positions of interest are represented by vertical lines and static objects by double vertical lines. When objects are linked together, such as the tools of a palette, they are linked by a double horizontal line. Object movements are depicted by diagonal lines. When two objects move together, such as the cursor and a dragged object, the two lines are linked by a single horizontal bar.

4.2 CIS Predictions and Previous Results

We have used SimCIS to compare techniques on a variety of sequences. Here, we report the results when using a toolglass and a fixed palette on sequences Seq_1 and Seq_2 (from Figure 3). Seq_1 minimises distances between work objects, while Seq_2 minimises the number of tool switches.

Figure 5 summarises the predictions computed by SimCIS: the fixed palette is highly sensitive to context for both time and action complexity. Figure 6 shows the OMD for the sequences Seq_1 and Seq_2 and the fixed palette and illustrates why Seq_1 is more complex than Seq_2 for this interaction technique: S_1 requires many round trips to the palette whereas Seq_2 does not. The toolglass shows no sensitivity to context in action complexity, and very little in time complexity. The fixed palette is more efficient for Seq_2 while the toolglass is more efficient on Seq_1.

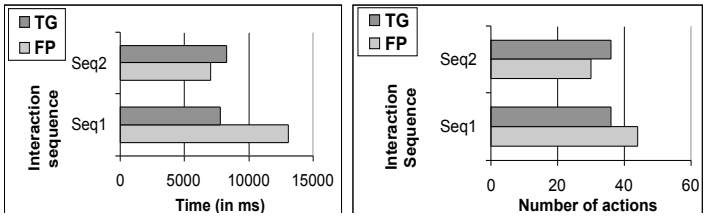
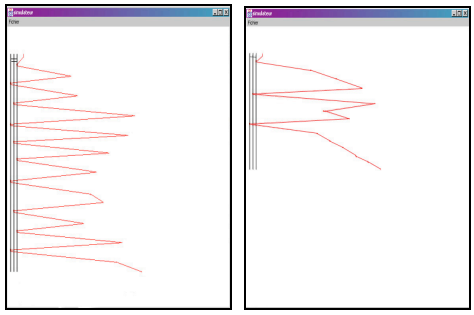


Figure 5: Comparing the complexity measures.



Sequence (size = 12)	Seq1	Seq2
Time (in ms)	13 056	7 015
Number of actions	44	30

Figure 6: OMD and palette complexity for S_1 (left) & S_2 (right).

These results are interesting because they confirm some of the experimental results reported in earlier work but challenge others. These sequences operationalise to some extent the problem solving and copy contexts defined by

Mackay (2002): in problem solving, users tend to create objects according to their thought process and exhibit more locality, i.e. create objects of different types that are close together (as in Seq_1); In a copy context, users can plan further ahead and tend to create objects of the same type together to minimise tool switches (as in Seq_2). Mackay (2002) reports that users prefer palettes in a copy context while they prefer toolglasses in a problem solving context. This matches our predictions if we assume that expert users prefer the most efficient technique. In another study, Kabbash et al. (1994) also compare toolglasses and palettes, but their task forces a tool switch at each step. They conclude that toolglasses are more efficient than fixed palettes. This matches our prediction for sequences with maximal tool switches (such as Seq1), but does not recognise that a different task (such as Seq2) would probably have given very different results.

4.3 Hypothesis on trade-off between context and efficiency

In order to achieve a goal, defined as a desired state of the interface, users can choose among multiple interaction techniques and interaction sequences. This choice is informed by the state of the interface, their knowledge of the available interaction techniques, and how many actions they can plan ahead. Some tasks, such as copying, allow users to plan far in advance while others, such as problem solving, are more incremental (Mackay, 2002). We define the *interaction context* as the combination of the current state of the interface and the amount of planning users can do. The former depends on the user's past actions while the latter depends on the task at hand and the next identified goal to be reached. We assume that the choice of interaction sequence is driven by the perceived efficiency of each possible path: once they know what they want to do, users try to do it as fast as possible based on their knowledge of the interface.

5 Validating CIS

In order to test the validity of CIS predictions and of our hypothesis, we ran a controlled experiment comparing the three interaction techniques on different sequences. The techniques are: Fixed Unimanual Palette (FP), Toolglass (TG), and Fixed Bimanual Palette (BP). Bimanual Palettes (BP), implemented in the CPN2000 interface (Beaudouin-Lafon & Lassen, 2000), use two hands and two cursors. Each hand controls one cursor: the non-dominant hand is used to select tools in the palette while the dominant hand selects work objects. We chose these techniques because of their physical similarity but different properties (Table 1).

	FP	BP	TG
Persistence	Yes	Yes	No
Parallelism	No	Yes	Yes

Table 1: characteristics of the three techniques.

5.1 Task

A set of shapes (green squares, blue triangles and red circles) was displayed. A trial consisted of deleting all the shapes, one after the other, as fast as possible.

Each interaction technique contained three tools that matched the three shape types. To delete a shape, the subject had to apply the tool displaying the shape's type onto that shape.

5.2 Experimental Factors

We used a 3x2x2x3 within-subject design. Factors were:

- Technique*: Fixed Palette (FP), Toolglass (TG) or Bimanual Palette (BP)*;
- Length*: 6, 18*;
- Grouping*: Grouped (G) or Distributed (D)*;
- Order: *Radial (R), Spiral (S) or Free (F).*

We used two interaction sequence lengths to test the effect of the ability to plan the action sequence. Table 2 shows the four different screen layouts associated with Grouping and Order. In all cases, shapes are organised radially along 3 lines of 2 shapes (length=6) or 6 lines of length 3 (length=18). When shapes are grouped (G), all the shapes along a line have the same type. When they are distributed (D), all the shapes along a line are different. This factor was used in combination with order (below) to operationalise the context of use.

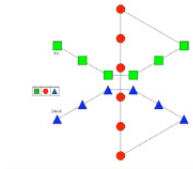
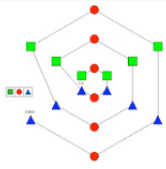
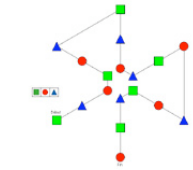
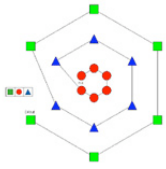
	R (Radial)	S (Spiral)
G (Grouped)	 Lo – Near	 Hi – Far
D (Distributed)	 Hi – Near	 Lo – Far

Table 2: the four types of imposed interaction sequences.

We imposed the order in which subjects had to delete the shapes for the R (Radial) and S (Spiral) trials, so as to test the validity of the predictions computed by SimCIS. The order was free in F trials, to test the hypothesis that subjects minimise execution time. For imposed-order conditions (R and S), subjects were asked to follow a black line showing the required deletion sequence. In the radial imposed order (R), shapes had to be deleted along each line; in the spiral imposed order (S), shapes had to be deleted along a spiral (Table 2).

The combination of grouping (G, D) and imposed order (R, S) defines four types of interaction sequences, classified as having low or high numbers of tool switches ('Lo' and 'Hi'), and short or long distances between successive objects ('Near' and 'Far') (Table 2). These factors correspond to a typical trade-off when planning a task: is it more efficient to optimise for distance between work objects at the

expense of more tool switches, or to optimise for tool switches at the expense of a longer distance between work objects.

5.3 Predictions and Hypothesis

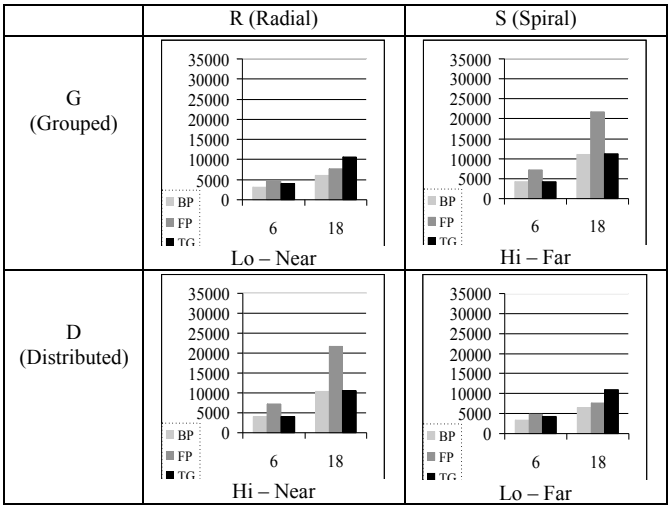


Table 3: SimCIS prediction for time complexity on imposed order trials.

Table 3 shows the time complexity predictions for the imposed order sequences as computed by SimCIS. We extract four predictions from this data that will be tested by analysing empirical data for the imposed order (R and S) conditions:

- ***(P_1)*** *FP and BP are highly sensitive to the number of tool switches ('Lo' vs. 'Hi') because of the persistence criteria.*
- ***(P_2)*** *BP is as fast or faster than the other two techniques because it exploits both persistence and parallelism.*
- ***(P_3)*** *Techniques are sensitive to object distance ('Near' vs. 'Far') because of the effect of Fitts' law on performance.*
- ***(P_4)*** *A longer sequence length exacerbates the difference between the worst and best cases because of the effect of planning.*

In the free (F) condition, subjects can choose in which order to do the task. The hypothesis underlying CIS is that subjects optimise for time according to the task and interaction techniques at hand. We will test this hypothesis by analysing empirical data for the free (F) condition:

- ***(H_{Free})*** *Subjects optimise for time, i.e., given a technique T, they plan a sequence of interactions that takes advantage of the characteristics of T.*

5.4 Subjects and apparatus

Twelve adult volunteers, 10 males and two females, all right-handed, signed up for 45 minute time slots. Ages ranged from 20 to 56 (mean=29.41, sd=9.73).

The training room contained one HP workstation XW4000 running Windows XP Professional, equipped with two WACOM tablets with one puck each for two-handed input. The right tablet was 145x125mm, the left was 456x361mm. The program was written in Java and ICON (Dragicevic & Fekete, 2001).

5.5 Procedure

The experiment consisted of 36 conditions grouped in three blocks, one block per technique. Each block consisted of three sub-blocks: a training sub-block to get familiar with the technique, a sub-block with free (F) trials, and a sub-block with imposed order (R and S) trials. The training sub-block was always first. The order of blocks and non-training sub-blocks within a block were counterbalanced across subjects using a 3x2 Latin square. The order of trials in a sub-block was counterbalanced within subjects. Each subject completed a total of 108 non-training trials (36 trials per technique, i.e. 3 repeated measures).

In the training sub-blocks, subjects had to delete shapes appearing one by one in the main window and clicked a button when they felt familiar with the technique. The next shape was always previewed at the top-right corner to allow subjects to plan tool switches (as in the popular Tetris game).

At the end of the experiment, each subject completed a survey. They were asked if they had previous knowledge of each technique, whether they had a preferred technique during the experiment and the cases in which each technique was preferred. They were also shown four free-order trials and asked to rank their preferred technique for completing each of them.

6 Results

Data was recorded at the trial level: time between first click and disappearance of the last shape, number of switches, and number of errors. We also recorded time between the disappearance of successive shapes. The Tukey HSD test was used for pairwise comparisons. Unless otherwise specified, data for the Length condition (6 or 18) is analysed separately.

6.1 Comparisons between Empirical Data and CIS Predictions

We start by comparing the empirical observations (Table 4) to the SimCIS predictions (Table 3). Although SimCIS underestimates the execution times, it predicts the pattern correctly, i.e. the relationship between interaction sequence and the efficiency of a technique. We next analyse the data from the imposed order conditions (R and S) to test our predictions.

Technique had a significant effect on execution time (length=6: $F_{2, 33} = 8.67$, $p = 0.0009$; length=18: $F_{2, 33} = 13.82$, $p < 0.0001$). Only pairs (BP, FP) and (BP, TG) are significantly different, so BP < TG ≈ FP. As predicted by CIS, *BP is more efficient than TG and FP* (Table 5) (**P_2**). Technique had no significant effect on number of errors (length=6: $F_{2, 33} = 1.42$, $p = 0.25$; length=18: $F_{2, 33} = 0.58$, $p = 0.56$), so differences between techniques cannot be explained by the number of errors. This is important since SimCIS does not take errors into account.

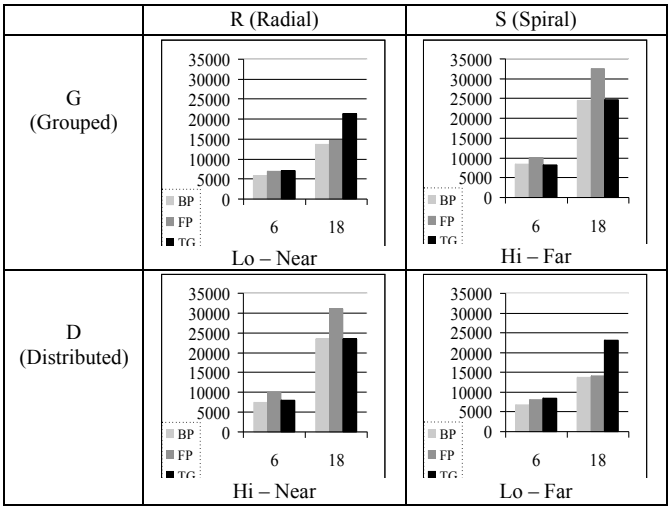


Table 4: Empirical mean time on imposed order trials.

	TG	BP	FP
length=6	7972	**7134**	8812
length=18	23142	**18903**	23100

Table 5: Mean execution times (ms).

Order interacts significantly with grouping on execution time for both palettes (length=6: $F_{2,\ 132} = 9.63$, $p = 0.0001$; length=18: $F_{2,\ 132} = 105.13$, $p < 0.0001$). As predicted, *both palettes are faster in conditions with low rather than high number of switches ('Lo', i.e. S→D and R→G vs. 'Hi', i.e. S→G and R→D)* (**P_1**). Also, *BP is less context sensitive than FP* as predicted by SimCIS (Table 3): differences between minimal mean time and maximal mean time are significantly larger for FP than for BP (length=6: $diff_{FP} = 3348$ and $diff_{BP} = 2728$; length=18: $diff_{FP} = 18465$ and $diff_{BP} = 10849$).

For each technique, *mean execution time is shorter in conditions 'Near' (short distance) than in conditions 'Far' (long distance)* (**P_3**) (Table 4). However differences do not reach significance (length=6: $F_{2,\ 66} = 0.15$, $p = 0.85$; length=18: $F_{2,\ 66} = 0.22$, $p = 0.8$). This may be due to the layout of the various interaction sequences: in the 'Near' condition, the distances are fairly high when jumping from one branch to the next, while in the 'Far' condition, distances become smaller when spiraling towards the inner circle of targets.

Table 6 shows the ratio between execution times for length=18 and length=6 sequences for each condition. We know that both palettes are sensitive to the number of tool switches. The table shows that this sensitivity increases with sequence length (**P_4**): the ratios are smaller in the 'Lo' (S→D, R→G) than in the 'Hi' (S→G, R→D) conditions. We also know that toolglasses are less sensitive to tool switches, and indeed the ratios are all close to 3 for TG.

Ratio	FP		BP		TG	
	G	D	G	D	G	D
R	2.12	3.08	2.40	3.13	2.98	2.94
S	3.20	1.74	2.91	2.02	2.96	2.75

Table 6: Ratio of mean execution time between length=18 and length=6 trials.

Kabbash et al. (1994) compared toolglasses to three other palettes, including R-tearoff menus (floating unimanual palettes) and L-tearoff menus (floating bimanual palettes). We were surprised by the poor performance of their equivalent to BP (*TG < FP ≈ BP*). This difference is not due to their use of floating rather than fixed palettes, since their subjects moved the BP in only 2.9% of trials.

If we consider trials with a low number of tool switches ('Lo' condition), our results show that TG is the worst: *BP ≈ FP < TG*. This is probably because FP becomes more efficient with fewer tool switches. The connect-the-dots task used by Kabbash et al. (1994) avoided this condition by forcing successive dots to differ in colour. However, we do not explain the difference between our results (*BP ≈ TG < FP*, predicted by CIS) for trials with a high number of tool switches ('Hi' condition, corresponding to the connect-the-dots task) and their results (*TG < FP ≈ BP*).

In summary, there is no such thing as the "best" interaction technique. Showing the advantages of a new technique is legitimate, but it is also important to link it to the context of use by studying worst-case scenarios, in order to obtain more generalisable results.

6.2 Subjects Optimise for Execution Time

The combination of imposed order and grouping was designed so that, for each technique, one was close to the optimal time. Subjects approach or even beat this time (Table 7) when they are asked to delete all shapes as fast as possible in the free (F) condition. We verified that there was no learning effect when the imposed order trials (R/S) were presented before the free ones (F) (length=6: $F_{1, 22} = 3.82$, $p = 0.0633$; length=18: $F_{1, 22} = 0.28$, $p = 0.6013$). The overhead of having to follow the black line in imposed-order trials may explain why free trials sometimes beat the best imposed-order trials.

	Length = 6						Length = 18					
	FP		BP		TG		FP		BP		TG	
	G	D	G	D	G	D	G	D	G	D	G	D
R	6846	10116	5722	7529	7187	7986	14542	31181	13711	23539	21385	23476
S	10194	8093	8450	6834	8312	8403	32570	14105	24560	13802	24632	23075
F	**6633**	**7128**	**5479**	**5946**	**7389**	**7907**	**14379**	**15796**	**13754**	**16224**	**21710**	**24095**

Table 7: Mean execution times (ms).

Analyses of the effect of technique on number of tool switches is not significant for length=6 trials ($F_{2, 33} = 2.67$, $p = 0.0836$) but is on length=18 trials ($F_{2, 33} = 64.56$, $p < 0.0001$, pairs (TG, BP) and (TG, FP) are significantly different). Subjects minimise the number of tool switches when they use a palette (FP or BP) in the 'Lo' condition, but not when they use TG (Table 8). This shows that subjects

understood how each technique was sensitive to the context and optimised its use accordingly ($\mathbf{H_{Free}}$).

	TG	BP	FP
length=6	3.23	3.12	2.80
length=18	**8.40**	3.11	2.00

Table 8: Mean number of switches in free order (F) trials.

In the post-hoc survey, only three subjects were able to describe in which trials a technique would be most efficient. One subject always preferred TG, three BP, and five FP, the latter arguing that they were more used to it. The answers to the final question contrast with these preferences yet are consistent with the quantitative data: subjects were asked to rank their preferred techniques to complete a free-order trial drawn on the survey; BP always scored better than the other two techniques, FP was better than TG on trials in condition G (Grouped) and TG was slightly better than FP on the 6-length trial in condition D (Distributed).

Altogether, these results show that users are able to optimise their use of an interaction technique and adapt it to the context at hand. Although they may not always be able to articulate the properties of interaction techniques, they are able to identify the most efficient technique for a given task. This both validates our hypothesis (at least on the techniques we have tested) and opens up new directions for the CIS model and the SimCIS tool.

7 Conclusion and Future Work

We have presented CIS, a model that describes the structure of interaction techniques and predicts the difference between their efficiencies in a given interaction sequence. We used it to predict differences of efficiency among three interaction techniques: fixed palettes (FP), bimanual palettes (BP), and toolglasses (TG). We conducted a controlled experiment to test these predictions: the efficiency of both palettes (FP and BP) is indeed more context-dependant than TG, and BP outperforms the other two techniques. The experiment also showed that subjects take advantage of this sensitivity to optimise execution time.

CIS is not intended to replace empirical evaluation but rather acts as a tool to help test multiple alternatives and design experiments. It can help explain the sensitivity of interaction techniques to context and identify best- and worse-case scenarios. We argue that, in order to be generalisable, experimental comparisons of interaction techniques should include the concept of context sensitivity.

We intend to develop CIS in several directions. First, we can improve the time complexity predictions by refining the model. For example, the largest differences between Tables 3 and 4 are due to toolglasses because we lack a proper model of double pointing. We plan to extend SimCIS to cover combinations of interaction techniques and automatic identification of best- and worse-cases. This is challenging due to the combinatorial explosion of the number of sequences to explore. Finally, we want to use CIS to help create new interaction techniques. One approach would be to infer possible interaction steps from a set of sequences and use these as a basis for an interaction technique.

REFERENCES

Amant, R. St. & Horton, T. E. (2002). Caracterizing Tool Use in an Interactive Environment. *Proc. Int. Symp. On Smart Graphics*, p. 86-93

Beaudouin-Lafon, M. & Lassen, H.M. (2000). CPN2000: A Post-WIMP Graphical Application.. *Proc. ACM Symposium on User Interface Software and Technology* (UIST'00). p. 181-190

Bier, E.A., Stone, M.C., Pier, K. & Buxton, W. (1993). Toolglass and Magic Lenses: the See-Through Interface. *Proc. ACM Siggraph*, p. 73-80

Card, S.K., Robertson, G. & Mackinlay, J. A. (1991). Morphological Analysis of the Design Space of Input Devices. *Proc. ACM Transactions on Information Systems*, 9(2), p. 99-122

Dragicevic, P. & Fekete J. D. (2001). Input Device Selection and Interaction Configuration with ICON. *Joint Proc. HCI'01 and IHM'01*. p. 543-558

Fitts, P. M. (1954). The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology*, (47), p.381-391

Foley, J. D., Wallace, V. L. & Chan, P. (1984). The Human Factors of Computer Graphics Interaction Techniques. *IEEE Computer Graphics & Applications*. p.13-48

Green, T.R.G. (2000). Instructions and Descriptions: some cognitive aspects of programming and similar activities. *Proc. ACM Working Conference on Advanced Visual Interfaces* (AVI'00), p. 21-28

Hick, W. E. (1952). On the Rate of Gain of Information. *Quarterly Journal of Experimental Psychology*, 4, p. 11-26

Jacob, R., Sibert, L., McFarlane, D. & Preston Mullen, M. (1994). Integrality and Separability of Input Devices. *Proc. ACM Transactions on Computer-Human Interaction*. 1(1), p. 3-26

John, B. E. & Kieras, D. E. (1996). The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. *Proc. ACM Transactions on Computer-Human Interaction*. 3(4), p. 320-351

John, B. E. & Kieras, D. E. (1996). Using GOMS for User Interface Design and Evaluation: Which Technique ? *Proc. ACM Transactions on Computer-Human Interaction*. 3(4), p. 287-319

John, B., Vera, A., John, B., Remington, R. & Freed, M. (2002). Automating CPM-GOMS. *Proc. ACM Human Factors in Computing Systems* (CHI'02), p. 147-154

Kabbash, P., Buxton, B. & Sellen, A. (1994). Two-handed Input in a Compound task. *Proc. ACM Human Factors in Computing Systems* (CHI'94), p. 417-423

Kieras, D.E. & Polson, P. G. (1985). An Approach to the Formal Analysis of User Complexity. *Int. Journal of Man-Machine Studies*, 22, p. 365-394

Mackay, W.E. (2002). Which Interaction Technique Works When? Floating Palettes, Marking Menus and Toolglasses support different task strategies. *Proc. ACM Conference on Advanced Visual Interfaces* (AVI'02). p. 203-208